Write programs to implement and compare the following sorting algorithms:

a. Bubble sort

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
   int a[10], i, j, t;

   // Input 10 values
   for (i = 0; i < 10; i++)
   {
      cout << "Enter value " << i + 1 << ": ";
      cin >> a[i];
   }

   // Bubble Sort algorithm (ascending order)
   for (i = 0; i < 9; i++)
   {
      for (j = 0; j < 9 - i; j++)
      {
         if (a[j] > a[j + 1])
         {
            t = a[j];
            a[j] = a[j + 1];
            a[j + 1] = t;
         }
```

```
        }
    }

    // Output sorted values
    cout << "\nAfter Bubble Sorting:\n";
    for (i = 0; i < 10; i++)
    {
        cout << a[i] << endl;
    }

    getch(); // waits for key press before closing console
    return 0;
}
```

Write programs to implement and compare the following sorting algorithms:
b. Insertion sort

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
    int a[10], i, j, key;

    // Input 10 values
    for (i = 0; i < 10; i++)
    {
        cout << "Enter value " << i + 1 << ": ";
```

```
    cin >> a[i];
}

// Insertion Sort algorithm
for (i = 1; i < 10; i++)
{
    key = a[i];
    j = i - 1;

    // Move elements of a[0..i-1] that are greater than key
    // to one position ahead of their current position
    while (j >= 0 && a[j] > key)
    {
        a[j + 1] = a[j];
        j--;
    }
    a[j + 1] = key;
}

// Output sorted values
cout << "\nAfter Insertion Sorting:\n";
for (i = 0; i < 10; i++)
{
    cout << a[i] << endl;
}

getch(); // Waits for key press before closing console
return 0;
```

```
}
```

Write a program to implement basic array operations: a. Insert an element at a specific position in an array.

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
   int a[20], n = 10, pos, value, i;

   // Input 10 elements
   cout << "Enter 10 elements:\n";
   for (i = 0; i < n; i++)
   {
      cin >> a[i];
   }

   // Input position and value
   cout << "\nEnter the position (1 to " << n + 1 << ") where you want to insert:
";
   cin >> pos;

   cout << "Enter the value to insert: ";
   cin >> value;
```

```
// Check for valid position
if (pos < 1 || pos > n + 1)
{
   cout << "Invalid position!";
}
else
{
   // Shift elements right
   for (i = n; i >= pos; i--)
   {
      a[i] = a[i - 1];
   }

   // Insert new value
   a[pos - 1] = value;
   n++; // Increase size

   // Display new array
   cout << "\nArray after insertion:\n";
   for (i = 0; i < n; i++)
   {
      cout << a[i] << " ";
   }
}
```

```
    getch(); // wait for key press before closing
    return 0;
}
```

b. Delete an element from a specific position in an array.

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{
    int a[20], n = 10, pos, i;

    // Input 10 elements
    cout << "Enter 10 elements:\n";
    for (i = 0; i < n; i++)
    {
        cin >> a[i];
    }

    // Input position to delete
    cout << "\nEnter the position (1 to " << n << ") of the element to delete: ";
    cin >> pos;
```

```
// Check for valid position
if (pos < 1 || pos > n)
{
   cout << "Invalid position!";
}
else
{
   // Shift elements left to fill the gap
   for (i = pos - 1; i < n - 1; i++)
   {
      a[i] = a[i + 1];
   }

   n--; // Decrease size after deletion

   // Display updated array
   cout << "\nArray after deletion:\n";
   for (i = 0; i < n; i++)
   {
      cout << a[i] << " ";
   }
}

getch(); // Wait for key press before closing
return 0;
```

```
}
```

c. Search for an element in an array (linear search).

```
#include <iostream>
using namespace std;

int main()
{
   int a[20], n, i, key;
   bool found = false;

   // Input number of elements
   cout << "Enter number of elements: ";
   cin >> n;

   // Input array elements
   cout << "Enter " << n << " elements:\n";
   for (i = 0; i < n; i++)
   {
      cin >> a[i];
   }

   // Input element to search
   cout << "Enter the element to search: ";
   cin >> key;
```

```
    // Linear search
    for (i = 0; i < n; i++)
    {
        if (a[i] == key)
        {
            cout << "Element found at position: " << i + 1 << endl;
            found = true;
            break;
        }
    }

    if (!found)
    {
        cout << "Element not found in the array." << endl;
    }

    return 0;
}
```

3. Write a program to implement a stack using an array.

```
#include <iostream>
#include <conio.h>
#include <alloc.h> // for older Turbo C++ memory functions
using namespace std;
```

```
struct Node {
    int rollNo;
    string name;
};

int main()
{
    Node stack[20];
    int top = -1;
    int choice;
    int roll;
    string name;

    do {
        cout << "\nEnter 1 for push, 2 for pop, 0 for exit: ";
        cin >> choice;

        switch (choice) {
        case 1:
            // Push
            cout << "Enter roll no: ";
            cin >> roll;
            cout << "Enter name: ";
            cin >> name;
```

```
    if (top == -1)
        cout << "1st node is created\n";
    else
        cout << "01 node is added\n";

    top++;
    stack[top].rollNo = roll;
    stack[top].name = name;

    // Display stack
    cout << "Current Stack:\n";
    for (int i = 0; i <= top; i++)
        cout << "roll no: " << stack[i].rollNo
             << " name: " << stack[i].name << " --> ";
    cout << endl;
    break;

case 2:
    // Pop
    if (top == -1)
        cout << "Stack Underflow!\n";
    else {
        cout << "Popped: roll no: " << stack[top].rollNo
             << " name: " << stack[top].name << endl;
        top--;
```

```
            }
            break;

        case 0:
            cout << "Exit\n";
            break;

        default:
            cout << "Invalid choice!\n";
        }

    } while (choice != 0);

    getch();
    return 0;
}
```

Write programs to implement and compare: a. Linear search

```
#include <iostream>
using namespace std;

int main() {
    int n, key;

    // Input array size
    cout << "Enter number of elements in the array: ";
```

```
cin >> n;

int arr[n];

// Input array elements
cout << "Enter " << n << " elements:\n";
for (int i = 0; i < n; i++) {
    cin >> arr[i];
}

// Input element to search
cout << "Enter the element to search: ";
cin >> key;

// Linear search
bool found = false;
int position = -1;
for (int i = 0; i < n; i++) {
    if (arr[i] == key) {
        found = true;
        position = i + 1; // +1 to show position starting from 1
        break;
    }
}
```

```
    // Display result
    if (found) {
        cout << "Element " << key << " found at position " << position << ".\n";
    } else {
        cout << "Element " << key << " not found in the array.\n";
    }

    return 0;
}
```

b. Binary search (on a sorted array)

C++

```
#include <iostream>
using namespace std;

int binarySearch(int arr[], int n, int key) {
    int first = 0;
    int last = n - 1;

    while (first <= last) {
        int middle = (first + last) / 2;

        if (arr[middle] == key) {
            return middle + 1; // position starting from 1
        }
        else if (arr[middle] < key) {
```

```
            first = middle + 1;
        }
        else {
            last = middle - 1;
        }
    }

    return -1; // not found
}

int main() {
    int n, key;

    cout << "Enter number of elements in the sorted array: ";
    cin >> n;

    int arr[n];
    cout << "Enter " << n << " elements in sorted order:\n";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    cout << "Enter element to search: ";
    cin >> key;
```

```
    int result = binarySearch(arr, n, key);
    if (result != -1) {
        cout << "Element " << key << " found at position " << result << ".\n";
    } else {
        cout << "Element " << key << " not found in the array.\n";
    }

    return 0;
}
```

C program:

```
#include <stdio.h>
#include <conio.h>

int binarySearch(int arr[], int n, int key) {
    int first = 0;
    int last = n - 1;

    while (first <= last) {
        int middle = (first + last) / 2;

        if (arr[middle] == key) {
            return middle + 1; // position starting from 1
        }
        else if (arr[middle] < key) {
```

```
            first = middle + 1;
        }
        else {
            last = middle - 1;
        }
    }

    return -1; // not found
}

int main() {
    int n, key;




    printf("Enter number of elements in the sorted array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements in sorted order:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter element to search: ");
```

```
    scanf("%d", &key);

    int result = binarySearch(arr, n, key);
    if (result != -1) {
        printf("Element %d found at position %d.\n", key, result);
    } else {
        printf("Element %d not found in the array.\n", key);
    }

    getch(); // wait for key press
    return 0;
}
```

5. Write a program to simulate a simple queuing system (e.g., customer service queue).

```
#include <iostream>
using namespace std;

#define MAX 5

int queue[MAX];
int front = -1, rear = -1;

// Add customer to queue
void enqueue(int customer) {
    if (rear == MAX - 1) {
```

```
        cout << "Queue is full!\n";
    } else {
        if (front == -1) front = 0;
        rear++;
        queue[rear] = customer;
        cout << "Customer " << customer << " added.\n";
    }
}

// Serve customer from queue
void dequeue() {
    if (front == -1 || front > rear) {
        cout << "Queue is empty!\n";
    } else {
        cout << "Customer " << queue[front] << " served.\n";
        front++;
    }
}

// Display queue
void display() {
    if (front == -1 || front > rear) {
        cout << "Queue is empty!\n";
    } else {
        cout << "Queue: ";
```

```
        for (int i = front; i <= rear; i++)
            cout << queue[i] << " ";
        cout << endl;
    }
}

int main() {
    int choice, customer;

    do {
        cout << "\n1. Add Customer\n2. Serve Customer\n3. Display Queue\n4. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch(choice) {
            case 1:
                cout << "Enter Customer ID: ";
                cin >> customer;
                enqueue(customer);
                break;
            case 2:
                dequeue();
                break;
            case 3:
```

```
            display();
            break;
        case 4:
            cout << "Exiting...\n";
            break;
        default:
            cout << "Invalid choice!\n";
      }
   } while(choice != 4);

   return 0;
}
```

6. Write a program to implement a queue using an array.

```
#include <iostream>
#include <conio.h>
#include <cstdlib>  // for malloc
using namespace std;

struct stud {
   int rno;
   char name[10];
   struct stud* n;
} *f = NULL, *r = NULL;

void enqueue() {
```

```
    struct stud* t;
    t = (struct stud*)malloc(sizeof(struct stud));

    cout << "Enter roll no: ";
    cin >> t->rno;
    cout << "Enter name: ";
    cin >> t->name;

    t->n = NULL;

    if (f == NULL) { // first node
        f = r = t;
        cout << "\n1st node is created\n";
    } else {
        r->n = t;
        r = t;
        cout << "\nNode is added\n";
    }
}

void dequeue() {
    if (f == NULL) {
        cout << "Queue is empty!\n";
        return;
    }
```

```
    struct stud* t = f;
    cout << "\nStudent removed: Roll no: " << t->rno << ", Name: " << t->name
<< endl;
    f = f->n;
    free(t);
    if (f == NULL) r = NULL; // queue became empty
}

void display() {
    if (f == NULL) {
        cout << "Queue is empty!\n";
        return;
    }
    struct stud* t = f;
    cout << "\nQueue:\n";
    while (t != NULL) {
        cout << "Roll no: " << t->rno << ", Name: " << t->name << endl;
        t = t->n;
    }
}

int main() {
    int ans;
    do {
        cout << "\nEnter 1 for push, 2 for pop, 3 to display, 0 for exit: ";
```

```
        cin >> ans;

        switch(ans) {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 0:
                cout << "Exiting...\n";
                break;
            default:
                cout << "Invalid choice!\n";
        }

    } while(ans != 0);

    getch();
    return 0;
}
```

Merge Sort

```
#include <iostream>
#include <conio.h>
using namespace std;

int main() {
   clrscr();
   int a[10], i, j, n = 10, t;

   // Input 10 values
   for (i = 0; i < n; i++) {
      cout << "Enter value " << i + 1 << ": ";
      cin >> a[i];
   }

   // Merge Sort logic (simple)
   int temp[10];
   for (int size = 1; size < n; size *= 2) {
      for (int left = 0; left < n; left += 2 * size) {
         int mid = left + size - 1;
         if (mid >= n) mid = n - 1;
         int right = left + 2 * size - 1;
         if (right >= n) right = n - 1;

         int i1 = left, i2 = mid + 1, k = left;
```

```
        while (i1 <= mid && i2 <= right) {
            if (a[i1] <= a[i2])
                temp[k++] = a[i1++];
            else
                temp[k++] = a[i2++];
        }
        while (i1 <= mid) temp[k++] = a[i1++];
        while (i2 <= right) temp[k++] = a[i2++];
        for (k = left; k <= right; k++)
            a[k] = temp[k];
      }
   }

   // Output sorted array
   cout << "\nAfter Merge Sorting:\n";
   for (i = 0; i < n; i++)
      cout << a[i] << endl;

   getch();
   return 0;
}
```

Tower of Hanio

```
#include <iostream>
#include <conio.h>
using namespace std;
```

```
void hanoi(int n, char from, char to, char aux) {
   if (n == 1) {
      cout << "Move disk 1 from " << from << " to " << to << endl;
      return;
   }
   hanoi(n - 1, from, aux, to);
   cout << "Move disk " << n << " from " << from << " to " << to << endl;
   hanoi(n - 1, aux, to, from);
}

int main() {

   int n;

   cout << "Enter number of disks: ";
   cin >> n;

   cout << "\nSequence of moves:\n";
   hanoi(n, 'A', 'C', 'B'); // A = source, B = auxiliary, C = destination

   getch();
   return 0;
}
```